AMENDMENT TO THE CLAIMS

This listing of claims will replace all prior versions of claims in the application.

Listing of Claims:

1. (currently amended) A processor comprising:

a queue configured to store one or more instructions to be issued; and

a control circuit coupled to the queue, wherein the control circuit is configured to detect a replay of an instruction in a load/store pipeline due to a load miss, and wherein the control circuit is configured to enter into a stall state and inhibit issuance of instructions ~~in the queue~~ to the load/store pipeline and to one or more other pipelines by comparing a destination register corresponding to the instruction causing the load miss to other instructions in the queue for dependencies to the load miss, until fill data in response to the load miss is ~~being~~ returned for loading, the control circuit further includes a storage device to store a miss tag corresponding to the destination register of the instruction causing the load miss and to compare the stored miss tag to a fill tag that is sent to the control circuit when fill data is returned, in which when the miss tag and the fill tag match, the control circuit to exit the stall state to start issuing instructions from the queue to the pipelines.

2. (previously presented) The processor as recited in claim 1 wherein the control circuit is configured to inhibit issuance of the instructions until fill data is provided to a data cache of the processor.

3. (canceled)

4. (previously presented) The processor as recited in claim 2 wherein the control circuit further includes a comparator coupled to the storage device to compare the miss tag to the fill tag to determine if the fill data being returned corresponds to the load miss.

5. (previously presented) The processor as recited in claim 2 wherein multiple load

misses may be present in which comparison of the miss tag and the fill tag identifies fill data to its corresponding load miss.

6-9. (canceled)

10. (previously presented) The processor as recited in claim 2 wherein the control circuit is configured to permit issuance of one of the one or more instructions if one or more instructions lack dependency to the load miss.

11. (previously presented) The processor as recited in claim 10 wherein dependencies to the load miss are maintained by one or more scoreboards coupled to the control circuit.

12. (previously presented) The processor as recited in claim 11 wherein the control circuit is configured to detect dependencies on the load miss using one or more scoreboards which track instructions that have passed a stage of a pipeline where replay is signaled.

13. (canceled)

14. (currently amended) A method comprising:

detecting a replay of a first instruction due to a dependency on a load miss in a load/store pipeline of a processor;

comparing a destination register corresponding to the instruction causing the load miss to other instructions in the queue for dependencies to the load miss;

inhibiting issuance of one or more instructions from a queue to ~~a pipeline of a processor~~ the load/store pipeline and to one or more other pipelines of the processor responsive to detecting the replay of the load/miss in the load/store pipeline by entering a stall state;

storing a miss tag corresponding to the load miss in response to detecting the replay;

generating a fill tag when fill data corresponding to the load miss is returned

comparing the fill tag to the miss tag to identify when fill data corresponding to the load miss is being returned; and

exiting the stall state to allow one or more instructions to issue to the pipelines after comparing the fill tag and the miss tag results in a match.

15. (previously presented) The method as recited in claim 14 wherein exiting the stall state to allow one or more instructions to issue occurs after fill data is provided to a data cache.

16. (canceled)

17. (previously presented) The method as recited in claim 15 further comprising reading the miss tag from a read queue that stores one or more load misses, wherein the miss tag identifies the load miss.

18. (previously presented) The method as recited in claim 15 wherein multiple load misses may be present in which comparing the miss tag and the fill tag identifies fill data to its corresponding load miss.

19-22. (canceled)

23. (previously presented) The method as recited in claim 15 further comprising permitting issuance of one of the one or more instructions if one or more instructions lack the dependency to the load miss.

24. (previously presented) The method as recited in claim 23 further comprising detecting lack of dependency for an instruction in one or more scoreboards.

25. (previously presented) The method as recited in claim 23 further comprising detecting lack of dependency for an instruction by checking one or more scoreboards which track instructions that have passed a stage of the pipeline where replay is signaled.

26. (canceled)

27. (currently amended) A computer accessible medium comprising one or more data structures to manufacture a processor, the processor including:

a queue configured to store one or more instructions to be issued; and

a control circuit coupled to the queue, wherein the control circuit is configured to detect a replay of an instruction in a load/store pipeline due to a load miss, and wherein the control circuit is configured to enter into a stall state and inhibit issuance of instructions ~~in the queue~~ to the load/store pipeline and to one or more other pipelines by comparing a destination register corresponding to the instruction causing the load miss to other instructions in the queue for dependencies to the load miss, until fill data in response to the load miss is ~~being~~ returned for loading, the control circuit further includes a storage device to store a miss tag corresponding to the destination register of the instruction causing the load miss and to compare the stored miss tag to a fill tag that is sent to the control circuit when fill data is returned, in which when the miss tag and the fill tag match, the control circuit to exit the stall state to start issuing instructions from the queue to the pipelines.